SCALABLE TREE STRUCTURED HIGH SPEED I/O SUBSYSTEM ARCHITECTURE

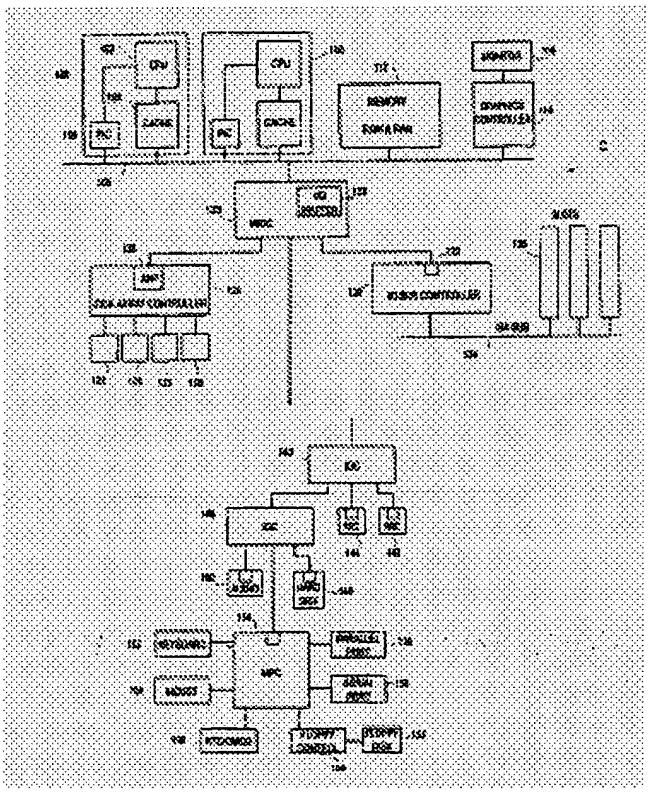
Patent number: WO9414121
Publication date: 1994-06-23
Inventor: WOOTEN DAVID R; MILLER CRAIG A; LEIGH KEVIN B; COSTLEY ROBERT BRETT; SIMONICH CHRISTOPHER E
Applicant: COMPAQ COMPUTER CORP (US)
Classification:
- international: G06F13/40
- european: G06F13/40D2
Application number: WO1993US11847 19931206
Priority number(s): US19930007333 19930121; US19920986918 19921208

Also published as:
EP0625274 (A1)
US5590292 (A1)

Cited documents:
US4100601
US4237447
WO9114229

Abstract of **WO9414121**

A point to point connection architecture for a computer I/O subsystem, resulting in a scalable tree structure. A Master I/O Concentrator (MIOC) is connected to the host bus and handles conversion between a bus oriented structure and the tree structure of the I/O subsystem. Ports away from the host bus are downstream ports and conform to a simple byt wide message protocol. Various IOCs and devices can be attached to one of the downstream ports on the MIOC. The MIOC directs transmissions to the appropriate channel based on a geographical addressing scheme. The IOC connections act as further points of branching. Ultimately IOD or I/O devices are reached, having an upstream port for connection to the IOC and a downstream port and internal logic appropriate for the particular peripheral device. Various registers are present in the IOCs and the IODs to allow determination of the topology and particular devices present. Messages and commands are transfered in the I/O subsystem in defined packets. Various read, write and exchange commands are used, with a read response being utilized to allow split transaction read operations. Certain status and control commands are also present. Interrupts are handled by having the interrupt levels correspond to memory addresses of the programmable interrupt controller, thus allowing simple selection of interrupts to be generated by the devices and no need for separate wiring.



SCALABLE TREE STRUCTURED HIGH SPEED I/O SUBSYSTEM ARCHITECTURE

Description of **WO9414121**

SCALABLE TREE STRUCTURED HIGH SPEED1/0
SUBSYSTEM ARCHITECTURE
The invention relates to computer system architectures, and more specifically to an architecture having input/output devices arrayed in a tree configuration from a main bus, with data transfer being done with command and data packets.

Personal computers have been developing very rapidly.

Initial designs had relatively low performance microprocessors, with relatively low performance input/output (I/O) and peripheral devices. Therefore a simple conventional bus architecture was quite adequate. However, performance of some of the components began increasing at a very high rate. Soon the simple bus architectures, particularly those with separatel/O spaces became a limiting factor. The bus speeds were simply too slow for adequate peripheral and I/O throughput.

Several variations were tried to improve the capabilities of the bus, mainly increasing the data path widths and transfer cycle speeds, but the bus architecture was still a limiting factor. Because interchangeable circuit boards were desired, widths were limited, as were speeds. Additionally, device loadings and capacitances became a problem, so that fewer slots were available at the highest of speeds. And yet the microprocessors continued to increase in performance, as did peripheral performance as increased use was made of local processors to allow parallel operation. But still the bus speed limitations remained. Variations were suggested that required removal of the card slots, but this solution provided only a short term solution, with the next generation of microprocessors again due to out strip this more integrated solution.Thus, while computer system performance was increasing, the effective rate of increase was significantly less than the basic processor performance improvement, system flexibility was being reduced and costs and complexities were being increased.

Further, the use of buses limited the number of available slots and layout of any slots. The number of slots available internally on a bus was practically limited to about eight due to electrical loading limitations. External expansion slots were not usually viable for high speed operations because of timing problems induced in the connection cabling. And the buses limited the layout alternatives of the slots. To be at all efficient of circuit board space the bus conductors had to run parallel, with the slots thus also being parallel, forming a rectangular box which had to be reserved for expansion cards.

When a design was started, this rectangular box had to be included as a requirement, greatly reducing the flexibility of the design. Additionally, concerns of signal skew due to varying length conductors and reflections due to multiple taps also necessitated the conventional physically parallel structure.

Notebook and handheld computers have become quite powerful. However, because of their small size, expansion of capabilities is very difficult. Historically, custom modules were required for each unit because of form factor concerns.

Recently, PCMCIA cards have become available. Their small size, approximately that of a thick credit card, has allowed their use in notebook computers. But again, expansion is still limited. Usually the maximum number of cards which can be incorporated is two, because of the size limitations incurred because of the bus connection used with PCMCIA. So even then notebook and handheld computer expansion is limited.

Therefore a new system architecture was needed which allowed for compatibility with existing software but allowed for a more performance improvement than a conventional bus architecture for the I/O devices while reducing costs, and provided greater expandability and used space more efficiently.

A system according to the present invention allows a computer to utilize existing applications software, but allows for a great improvement in1/0 capabilities at a lower cost than present solutions, with greater expansion capabilities and.

lower space requirements. The basis for the system is a pointto-point connection for the I/O subsystem, resulting in a tree structure, allowing parallel I/O operations separate from processor operations and separation of I/O devices with varying bandwidths. By reducing the loadings on the transmission lines to a single load, very high data rates can be achieved with MOS logic.

A Master I/O Concentrator (MIOC) is connected to the conventional, high performance host bus and handles conversion between a bus oriented structure and the tree structure of the I/O subsystem. Ports away from the host bus are downstream ports and conform to a simple byte wide transmission protocol to further allow high speed transfer. Various IOCs and devices can be attached to one of the downstream ports on the MIOC.

The MIOC directs transmissions to the appropriate channel based on a geographic addressing space. The IOC connections act as further points of branching. Ultimately IOD or I/O devices are reached, having an upstream port according to the protocol of the present invention and a downstream port and internal logic appropriate for the particular peripheral device of interest.

Various registers are present in the IOCs and the IODs to allow determine of the topology and particular devices present.

Messages and commands are transferred in the I/O subsystem in defined packets. Various read, write and exchange commands are used, with a read response being utilized to allow split transaction read operations. Certain status and control commands are also present. Interrupts are handled by having the interrupt levels correspond to memory addresses of the programmable interrupt controller, thus allowing simple selection of interrupts to be generated by the devices and no need for separate wiring.

In certain multiprocessor variations, the processor cards each include an MIOC and communicate directly with each peripheral group and shared memory, thus not having a common, shared bus which would reduce performance. In a small system variation, the MIOC is located directly on the microprocessor chip, with ports for the graphics/video system and to an IOC through which the #remaining peripherals are accessed. The. memory system is connected to the microprocessor and addressable by the internal MIOC. Thus the actual connection of the I/O subsystem to the processor and memory may vary from design to design and numerous other examples and connections can be developed.

The reduced number of interface pins allows the physical size of an A-Net-based device to be very small, as the connector requirements are minimal. This lends itself well to slotted or connected arrangements. These small size and slot capability allows greater expansion capabilities for notebook computers and easier connection to an expansion base. Further, the small size lends itself to a simple, daisychain stacking arrangement, increasing device interconnect flexibility.

Alternatively, the small size removes the circuit board layout problems as only a very few wires are needed, allowing slots to be placed where they best fit, not all massed together.

Thus a high performance, expandable, yet low cost,I/O subsystem architecture is provided according to the present invention.

A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:
Figures 1A and 1B are a block diagram of a computer system utilizing the architecture of the present invention;
Figures 1C and 1D are a block diagram of an alternate embodiment of a computer system utilizing the architecture of the present invention, with external expansion capabilities indicated;
Figure 2 is a schematic diagram of a connection between input/output communication ports in the computer system of
Figures 1A and IB;
Figure 3 is a block diagram of an input/output communication port as used in the computer system of Figures1A and IB;
Figure 4 is a block diagram of an input/output concentrator of Figures1A and IB;;
Figures 5 and 6 are register maps for input/output concentrators and devices of Figures 1A and IB;
Figures 7A, 7B and 8 - 12 are timing diagrams of operations between two input/output communication ports according to the present invention;

Figures 13A and 13B are flowcharts of the initialization sequence for the input/output units of Figures 1A and IB;
Figure 14 is a block diagram of a multiprocessor system without a shared bus;
Figure 15 is a block diagram of a very simple system;
Figure 16A is a block diagram of a daisy-chain connection scheme;
Figure 16B is an exploded, perspective view of circuit boards of the daisy chain of Fig. 16A;
Figure 16C is a top view of a circuit board of Fig. 16B; and
Figure 16D is a bottom view of a circuit board of Fig.

16B.

Prior to addressing the figures, it is considered helpful to define certain terms. A node is a device having at least one electrical means of sending or receiving information. A channel is the connection between nodes. A port is the electrical interface in a node that comprehends a channel protocol. A point-to-point channel is a channel to which only two nodes are connected. A field is a collection of bits and/or bytes that, taken together, convey a single piece of information. A packet is any grouping of one or more fields.

A message is a packet containing a command field, or a command field followed by an address field, or a command field followed by a data field, or a command field followed by an address field followed by a data field. A packet will generally mean a packet with the attributes of a message. Transmission is sending a message on a channel. Upstream is referenced as being towards system memory, while downstream is considered to be away from system memory. An IOC is an input/output concentrator, which is a device having at least one upstream port (usually only one) and at least one downstream port (usually more thanone). The function of an IOC is to control the movement of messages between the upstream port(s) and the downstream port(s). The protocols on anIOC's upstream port(s) and downstream port(s) may differ.An MIOC is an IOC that has an upstream port interfacing to system memory. An IOD is an input/output device having a downstream port connected to a peripheral device. A device can combine properties of an IOC and an IOD to include a downstream port to a peripheral device and a downstream ANP. A collision is an attempt by both ports to arbitrate for an idle channel at the same time. A-Net is the phrase used to generally refer to components conforming to the present invention.

Referring now to Figures 1A and 1B, a computer system C incorporating the architecture of the present invention is shown. The illustrated embodiment is a multiprocessor configuration. It is understood that the architecture would also apply in single processor cases. A first processor 100 includes a CPU or microprocessor 102 which is connected to a cache system 104, with the cache system 104 in turn being connected to a main or host bus 106. A programmable interrupt controller(PIC) 108 is connected between the host bus 106 and the microprocessor 102, with this connection to be described below. A second processor 110 includes similar components and is connected in a similar fashion as the processor 100. The main memory 112 which includes read only memory (ROM) and random access memory (RAM) is connected to the host bus 106.

Preferably the memory 112 includes sufficient memory controllers and memory decode logic to allow access.

Additionally, a graphics controller 114 is connected to the host bus 106. The host bus 106 is preferably a very high speed bus according to the prior art and would, for example, have a data bus portion which is 64 bits wide having very short cycle times. This will allow a fast movement of data along the bus 106. The graphics controller 114 has a monitor 116 connected to provide a video output.

In a conventional system, a bus controller would then be used to link from the host bus 106 to aninputjoutput bus, such as the EISA or MCA buses. However as noted in the background of the invention, such-buses are highly bandwidth limited and so in the computer system C the input/output and peripheral devices are connected in an entirely different manner, in a point to point connection scheme resulting in a tree structure.

An MIOC 120 is connected to the host bus 106. The MIOC 120 preferably includes an I/O mapper 122, which performs the function of mapping or translating I/O space addresses as provided by Intel Corporation processors to a memory mapped environment, as the architecture of the computer system C preferably is a flat, memory mapped architecture with no split as conventional in Intel processors. By use of the I/O mapper 122, conventional personal computer software can be executed on the computer C without a need to recompile or reassemble the applications programs to reference the peripheral devices

at memory locations instead of the prior and conventional I/O locations. Details of the MIOC 120 will be described below.

The MIOC 120 has three downstream ports which are configured as
A-Net ports (ANPs) according to the present invention. Many details of the A-Net ports will be described below. The MIOC 120 can contain a number of ports but in the present example three are shown. One downstream port is connected to a disk array controller 124 which includes an upstream ANP 126 for connection to the MIOC 120. The disk array controller 124 then performs the various functions needed to control a plurality of hard disk drives 128 in fashions as known in the prior art and to those skilled in the art. This arrangement is preferred in the computer system C because the disk array controller 124 is a high performance unit which has a very large data throughput capability and so a direct connection to the MIOC 120 is preferred.

Additionally, an I/O bus controller 130 has an ANP 132 connected to the downstream ports of the MIOC 120. The bus controller 120 on its downstream port preferably provides an
ISA bus 134 which has a plurality of slots 136. Preferably the bus 134 is ISA and not one of the more powerful EISA or MCA buses as it is used only for certain compatibility reasons, with the majority of the high performance peripheral devices which would utilize the capabilities of the more advanced buses being connected according to the architecture of the present invention. Indeed in m#any cases it maybe. desirable not to even have an I/O bus controller 130, but merely to include additional slots connected to various ports of various IOCs.

Alternatively, the graphics controller 114, and/or a live motion video system, could be connected to the MIOC 120 instead of being connected to the host bus 106.

An IOC 140 has its upstream port connected to the final downstream port of the MIOC 120. The IOC 140 in the shown example has three downstream ports, two of which are connected to network interface cards 142 and 144. Preferably the NIC cards 142 and 144 also contain ANP ports to allow direct connection to the IOC 140. The final downstream port of the
IOC 140 is provided to the upstream port of yet another IOC 146. The IOC 146 preferably has three downstream ports, with one being connected to a hard disk unit which has a compatible upstream ANP. A second is connected to an audio interface 160.

The third port of the IOC 146 is connected to an ANP port of a multiple peripheral chip (MPC) 152. Preferably the MPC 152 is a combination chip as well known in the personal computer system design and includes interfaces to a parallel port 156, a serial port 158, a keyboard 162, a mouse or pointing device 164, real time clock/CMOS memory 166 and a floppy disk controller 150 which has connected to it a conventional floppy disk drive 152.

As can be seen from the architecture of the computer system C, the lowest performance peripherals are connected to the MPC 154, which is three levels of concentrator, that is the
MIOC 120, IOC 140 and thelOC 146, away from the host bus 106.

This fan out or tree structure is acceptable as the reduced data requirements of the devices reduces as the tree is traversed to a lower level. For example, the hard disk 148 and the audio system 160 connected to the IOC 146 are higher performance devices and so are connected at a higher level of bandwidth and capacity. Similarly, the network interface cards 142 and 144 are considered to be very high bandwidth devices and thus preferably are connected further up the tree.

Referring now to Figures 1C and 1D, an alternate embodiment of a computer C' is shown. Where elements are similar to those of the computer C shown in Figures 1A and 1B, like numbers are repeated. The first difference of the computer C' is that the MIOC 120' has seven ports to downstream devices to allow seven A-Net devices to be connected to it.

The first device of interest is the graphics controller 500, which is now connected to the MIOC 120'. It is further noted that two A-Net channels are utilized in this connection. This allows a higher bandwidth between the two devices, which is particularly appropriate for a graphics controller 500 which can utilize this higher bandwidth. The graphics controller 500 has connected to it a frame buffer 502 which it controls and which is used to store the actual pixel information which is provided to the monitor 116. A video controller 504 is also connected to the graphics controller 500. The video controller has video input and output ports and can receive an antenna 506. The video controller 504 is used with live motion video from

such sources as video recorders, video cameras, and broadcast television.This information is directly provided to the graphics controller 500, preferably via an A-Net channel.

This allows a simple interface to be developed in both units.

Further, by connecting the video controller 504 to the graphics controller 500 through an A-Net port, thus having the graphics controller to some extent operate as an IOC, then the bandwidth required by the video controller 504 to graphics controller 500 link is effectively isolated from the host bus 106 and from the MIOC 120'. This allows a greater system performance than if the units were directly connected to the host bus 106.

A second difference is the connection of a serial IOC 508 to the MIOC 120'. The serial IOC 508 in this case has a downstream serial channel which conforms to the A-Net command and packet protocol except that all data transfers are done in a serial manner. This allows for easy expansion to a location remote from the computer C'. For example, an expansion unit EI can contain a second and matching serial IOC 510. The serial
IOCs 508 and 510 are a matching pair, one having an upstream serial port and one having a downstream serial port. The serial IOC 510 can have connected to it for example, memory 512, 514 and 516 so that the expansion box EI can be a memory expansion box, for example.Given the bandwidth of the A-Net channel at a preferred data rate of 50 Mbytes per second, assuming that this memory is somewhat infrequently accessed and has a relatively high locality so that necessary data is commonly cached, memory expansion in this manner is quite easy.

The memory 512, 514 and 516 could for example be nonvolatile memory such as EEPROM used for long term storage.

An IOC 518 is connected to the MIOC 120'. This IOC 518 also has illustrated seven ports for connection to downstream devices. Five of the ports are connected to A-Net slots 520.

The A-Net slots 520 are used to receive interchangeable cards which include an A-Net port in an interface as will be described below. Because only a small number of pins are utilized in an A-Net channel, the A-Net slots 520 can be quite small, such as for example 28 pins, the 16 extra pins being used for power and ground connections. This allows a small connector size, which allows the system designer great latitude in the physical slot locations. Thus parallel connections as in typical slots are not required and system flexibility is increased. One of the slots 520 could contain a board including a serial IOC 522. The serial IOC 522 could be connected to a mating serial IOC 524 in a second expansion box
E2. The serial IOC 524 could be connected to a downstream IOC 526 and several devices 530. The IOC 526 may also have connected to it a further series of A-Net slots 528.Thus the number of A-Net slots can be easily and simply increased by use of an additional expansion box. It is considered desirable that the computer C' contains a minimum number of A-Net slots 520 necessary to meet the requirements of the greatest number of users, while an expansion unit E2 can be utilized with a relatively small performance degradation to meet the needs of a larger number of users which need more slotted expansion capabilities.

In certain very low bandwidth cases, a single downstream port on an IOC 518 can be shared by several I/O devices 532.

In that special case device selects such asDSO*, DS1* and DS2* also are provided from the IOC 518.

Referring now to Figure 14, an alternate multiprocessor configuration is shown. Processors 600, 602 and 604 are shown, with each processor having essentially identical configurations. An exemplary processor design is shown for the processor 600. The processors. include a CPU 606 which is connected to a local memory 608 and to a local MIOC 610. The
MIOC 610 has a number of upstream ports which are connected to the various peripheral blocks to be explained. This is preferably a relatively loosely coupled multiprocessor system.

In the illustrated embodiment, three basic peripheral blocks are shown. The first is the disk peripheral system 612. An
IOC 614 has three downstream ports in the illustrated embodiment to be connected to an A-Net port from each of the processors 600, 602 and 604. Obviously, if more than three processors were used, more than three processor-connected ports would be present in the IOC 614. In the embodiment shown, the

IOC 614 has ports connected to disk controllers 616 and 618, which may have SCSI buses or other standard configurations developed from them. This arrangement allows each processor 600, 602 and 604 to independently address the disk subsystem 612. No bus sharing is required or bus arbitration is required as no shared bus is necessary for the processors to access the disk sybsystem 612.The IOC 614 does all necessary port arbitration, allowing an effective high throughput to the disk subsystem 612 for each of the individual processors.

Preferably some shared memory also exists in the system.

This is shown as block 620 which has an upstream A-Net IOC 622 which has three downstream ports. The IOC 622 is connected to the shared memory 624 and thus includes the inverse of a MIOC in that it converts from A-Net to an upstream port having a memory interface. Giving the high data rates of the A-Net channel, particularly if multiple frequency clocks are utilized, the shared memory accesses need not result in effective slowing down of the system, particularly if the local processor memory 608 is sufficiently large.

The final basic subsystem illustrated is a terminal concentrator 622. An IOC 624 has ports connected to each of the processors 600, 602 and 604 and has ports connected to terminal concentrators 626, with each of the terminal concentrators 626 having the desired number of terminals connected thereto. Obviously other portions of the peripheral system could be connected in a like manner or the variousIOCs present in the subsystems could have additional channels so that additional subsystems could be developed. Thus it is seen.

in this system no bus is shared by the processors in the multiprocessor system, thus increasing system performance as arbitration for that shared bus is not necessary. Arbitration would only occur at a local device level and as this is believed to be greatly reduced as diversity of the operations increase overall system performance will also be increased.

In contrast to the complex system of Figure 14, a very simple computer system is shown in Figure 15. Figure 15 would be exemplary of a simple notebook or laptop computer. A microprocessor 700 is at the core of the system. The microprocessor 700 includes all necessary elements to be a processor, such as a CPU and other elements. The microprocessor 700 also includes a memory interface to connect to system memory 702. Further, the microprocessor 700 contains an A-Net port to connect to a graphics controller 704, with a monitor 706 connected to the graphics controller 704. A second
A-Net port is provided on the microprocessor 700 to connect to an IOC 708. The necessary remaining devices in the computer system, such as any additional IOCs needed for fan out and the particular devices, are connected below the IOC 708.The microprocessor 700 contains any necessary logic to allow the graphics controller 704 or the IOC 708 to access the memory 702. This could be done by a conventional busing arrangement with arbitration. It could also be done by various other known means, such as crosspoint switching and so on.

It is noted that the computer systems C and C' are exemplary and many other alternatives and various designs could be developed as will be apparent from the more detailed description of the various components of the architecture described below.

Referring now to Figure 2, the connection between two Anet ports is shown. The first A-net port 180 is considered to be the upstream port, while the second port 182 is considered to be a downstream port. There are twelve signals for each A
Net port. All signals are positive logic signals except those followed by an asterisk (*) which denotes negative logic.

signal Direction Descriptions
D < 7:0 > Input/Output Eight pins used to transfer
message fields (i.e. command,
status, address and data.)
PAR Input/Output Odd data parity bit of D < 7:0 > .

RTS* Output Request-to-Send.

CTS Input Clear-to-send.

CLK Output Clock output from the upstream port that is used to synchronize transfers on the channel.

Input Clock input to the downstream port.

Preferably each A-Net port is developed in a MOS technology and uses reduced-voltage-swing, open-drain drivers such as Gunning transistor logic for signaling between devices.

As noted above, A-Net slots can be developed. Figure 2 shows the preferred location of the connectors 181 and 183 with respect to the pullup resistorsRT. An A-Net slot would also include provisions for power and ground. As can be seen, with power and ground only 28 pins are needed for an A-Net slot.

This provides numerous advantages. The easier location of the slots in a system has been mentioned. Because of the reduced pin count on an A-Net board, the physical size of the board can be greatly reduced, to a size much smaller than even a PCMCIA card. The preferred size is in the range of a few square inches. This allows room for an A-Net slot connector, any necessary circuitry and a connector to an external device if necessary.

This small size lends itself quite nicely to use in a notepad or handheld computer. Four A-Net boards could easily be inserted in the space of two PCMCIA cards, thus effectively doubling the expansion capabilities of the notebook or handheld computer. The computers could be much more readily tailored to individual requirements. One of the A-Net slots could. be used for connection to an expansion base. The expansion base peripheral devices would readily merge into the A-Net system topology and the interface would require far fewer pins (and thus a much less expensive connector) than used in current expansion bases where effectively an ISA connector and additional pins are necessary.

Figures 16A, 16B, 16C and 16D illustrate one possible arrangement which can be developed because of the very limited number of pins necessary for an A-Net channel. Fig. 16A illustrates a block diagram of a daisy chain arrangement. A first module 750 includes an IOC 752 having one downstream and one upstream port and one connection to a peripheral device 754. A second module 756 has an IOC 758 whose channel downstream port is connected to the channel upstream port of the IOC 752 and which has a channel upstream port and a connection to a peripheral device 760. A third module 762 has an IOC 764 whose channel downstream port is connected to the channel upstream port of the IOC 758 and which has a channel upstream port for further chaining and a connection to a device controller 766. The device controller 766 is connected to an external peripheral device 768.It is noted that power and ground connections are preferably passed through each module, with each module tapped into the signals.

Illustrative modules 756 and 758 are shown in Fig. 16B.

The module 756 includes female connectors 800 preferably located on the bottom side of the module 756 and at each end.

Male connectors 802 are present on the top side of the module 756 to mate with female connectors 804 on the module 762. The female connectors 800 are the channel downstream side of the module 756, while the male connectors 802 are the channel upstream side of the module 756. The module 762 further contains a peripheral device connector 808, for example, on the same side as the female connectors 804. The allowable size of the connector 808 is such that room remains for the necessary components, such as an IOC and device logic. Male connectors 806 are provided to continue to stacking capability. Figures 16C and 16D are top and bottom views of the module 762 to allow better visualization.

Of course, other connector arrangements could be used, but in any case a very compact daisychaining arrangement would result. This allows a very high function packing density, allowing even smaller computers to be developed, which at the same time are very flexible.

With the narrow data path width and high clock rate, packets can be transferred at a very high rate, with timing details to be described. Preferably one byte is transferred every clock cycle for a base raw bandwidth of 50 MB/sec. The
A-Net message protocol is optimized for I/O applications. It supports all of the basic operations necessary for devices to respond to read and writes that are initiated by the system processors. These same

capabilities are available to the I/O devices so that they can initiate or master their own operations. Additional capabilities are included to the minimum set to optimize the overall efficiency of I/O in a system context. The message protocol has been extended where necessary to enhance error detection.

An A-Net message is classified as either a Data Movement or a Control/Status message. Within the Data Movement class there are seven message types. The format for each message is determined by its class and type as follows:
ClassTv#e Format
Data Movement read request < command+size >
(Read) < address >
n read request < command+size >
sequential (ReadS) < address >
Il write request < command+size >
(Write) < address > < data >
Il write request < command+size >
sequential < address > < data >
(WriteS)
compare and < command+size >
swap (CmpSwp) < address > < data >
exchange (Exch) < command+size >
< address > < data >
n read response < command+size >
(Response) < data >
Control/Status < command+subcommand >
The message class and type is used to determine the high order three bits (bits 7:5) of the first message byte, the
Command Byte. For all Data Movement commands, the low order five bits (bits 4:0) of the Command Byte contain a Size Field.

A packet's Size Field specifies the number of data bytes (1 to 32) to be transferred as a result of this command with the value encoded as one less than the actual number of bytes. For
Control/Status messages, the low order five bits of the Command
Byte are used for sub-command encoding.

Command Byte encoding is:
bit
7 6 5 4 3 2 10 Command
0 00 s s s s s Response
0 0 1 s s s s s Write
0 10 s s s s s WriteS
0 1 1 s s s s s Exchange
1 00 s s s s s Compare and Swap
1 0 1 s s s s s Read
1 10 s s s s s ReadS
1 1 1 1 0 0 00 Channel Error
1 1 1 1 0 0 0 1 Read Error
1 1 1 1 0 0 1 1 Channel Reset
1 1 1 1 0 1 00 Address Size 32
1 1 1 1 0 1 0 1 Address Size 40
1 1 1 1 0 1 10 Address Size 48
1 1 1 1 0 1 1 1 Address Size 64
1 1 1 1 1 0 0 O Little Endian
1 1 1 1 1 0 0 1 Big Endian
1 1 1 1 1 0 10 Idle
1 1 1 1 1 0 1 1 Request Retry
1 1 1 1 1 1 00 Real Time Command
1 1 1 1 1 1 1 1 Illegal Command
where sssss in bits 4 through 0 denote the Size Field.

When a message contains an address, the address is the next n bytes, the Address Bytes, of the message after the

Command Byte. The Address Bytes are sent in big-endian order with the first address byte being the most significant. The address size can be different in different systems but must be specified before any data transfers. Four, five, six or eight byte addresses can be used, with specification required before any data transfers.

Addresses are used for routing of messages through various
A-Net topologies. A message sent downstream on an A-Net channel will always be for the downstream device that receives the message or for devices attached to that downstream device.

Addresses sent upstream are usually intended for accesses to system memory or for signaling interrupt/exception conditions, as described below,
Each A-Net device is expected to have at least one addressable register. The number of addressable locations in a device is dependent on device and system implementation. All
A-Net topologies have a minimum of 4 KB of address space for each device. Another statement of this rule is that A-Net devices are addressable on 4 KB boundaries.

Messages containing data will have one or more bytes of data, the Data Bytes, following the Command Byte or, when present, the Address Bytes. The number of Data Bytes present in the message will be determined by the Size Field of the
Command Byte. Data Bytes are always present on Write, WriteS,
Exch, and Response messages. Consecutive Data Bytes have implied ascending address order.

An IOC or MIOC will not generally reorder messages received from a channel. Messages received on a downstream port will be sent to its upstream port in the order in which they were received. The IOC may send messages received from different downstream ports to its upstream port(s) in any order.

The Read and ReadS commands are used by downstream devices to read from system memory or by upstream devices to read from
I/O devices. Together they are referred to as Read Requests.

The size field in the Read Request Command Byte indicates the number of bytes that are to be returned to the device issuing the read. The Command Byte for a Read Request is followed by the Address Bytes. Data returned in response to a Read Request is denoted by a Response Command Byte. If an error is detected in reading the addressed data it is indicated to the reader with a single byte Read Error Command Byte. The ReadS command indicates that the next command will be a Read Request command to the sequential address after the present command, to allow for data prefetching.

Read Requests are split transactions allowing a channel to be used for other messages during the time that the requested data is being read from a device or from system memory. The limit on the number of outstanding Read Requests is system dependent. An attempt to issue more Read Requests than the implementation allows is indicated to the reader by a Retry during the Read Request that exceeds the implementation limit.

Additionally, a Read Request command requires the MIOC to receive a completion status on the system bus before issuing another command from the same channel.

The Write and WriteS commands are used by downstream devices to write to system memory or by upstream devices to write to I/O devices. Together they are referred to as Write
Requests. The size field in the Write Request Command Byte indicates the number of bytes that are being written. The
Command Byte for a Write Request is followed by the Address
Bytes and the number of Data Bytes indicated in the Size Field.

The WriteS command is similar to the ReadS command in that it indicates that the next command will be to a sequential address. Also a Write Request command is similar to a Read
Request command in that it requires the MIOC to receive a completion status on the system bus before issuing another command from the same channel.

The Exch command provides a mechanism for coordination of activities in multi-threaded environments. A-Net devices are semi-autonomous devices generally having access to main memory.

Once an I/O operation is initiated by a controlling processor, the A-Net device will begin a thread of operations to complete the I/O operation(s) that is asynchronous to the controlling processor's activities. Re-synchronization is accomplished with either interrupts or with Exch operations.

The A-Net Exch operation is defined to be a read-modifywrite that is atomic at the addressee. The Exch command format is the same as for Write Requests. It also has the attributes of a Read Request in that it requires a Read Response or Read
Error to the initiator. When an Exch command arrives at the addressee, the data is read and returned to the requestor, and the Exch data is then written to the addressed location.

The Cmpswp command is used to denote a write operation that will also return data. When an IOC receives this command, it should expect a Response to come from its upstream device.

A CmpSwp may be followed by 2, 4, 8, 16, or 32 data bytes and the address of the write must be aligned to a word boundary.

The data sent with the CmpSwp command are two equal sized operands. The first operand is a comparand and the second operand is a replacement. value. A device receivinga. C#p$wp command will read the contents of the addressed location and return that value in a Response. Then the device will compare this value to the comparand. If they are equal, the second operand is written to the addressed location. If they are different, the address location remains unchanged. If an odd address is detected or, if the size field is other than 2, 4, 8, 16 or 32, the device should return a Read Error command.

A device not supporting the CmpSwp command should return a Read
Error on receiving a CmpSwp command.

A Response message contains the data that was requested by a previously issued Read Request, Exch or Cmpswp command, collectively Requests. All responses sent downstream are provided in the order in which the Requests were issued upstream, with the converse case being also true. Upstream and downstream Requests are independent with no ordering required for the Requests in different directions. The IOC includes a read response FIFO queue to maintain this ordering. When an
IOC receives a Read or ReadS command, the IOC places the port number issuing the command into the queue. When a Response is received by the IOC, the first port number is popped from the
FIFO and the Response is forwarded to that port.

If an upstream device detects a fault condition on its upstream port, it will notify all downstream devices connected to it by issuing a Channel Error message. Any device having downstream ports will send Channel Error to those downstream ports when it receives a Channel Error from its upstream port.

If an addressed device. is unable to provide valid data in response to a Request, it will return a Read Error message to the channel from which the Request was received.

The Request Retry command is used by a device as a response to a request when the device is unable to respond in a timely fashion. If a device cannot send a Response within 256 channel clocks after a request is received, it must issue a Request Retry command.

The Idle command is used by an IOC to signal a channel with a pending request that the IOC is still waiting for a
Response. An IOC must respond with either an Idle command or a Response no later than 256 channel clocks after a request is received. If an Idle command is sent, the IOC must again send an Idle command or a Response within 256 channel clocks.

Another command is the Real Time Command. This is a prefix command, which means that it precedes another command byte, and is used to indicate that this operation relates to real time or urgent data transfer. This command is used to increase the priority of the channel responding to the command out of sequence so that real time data transfers are expedited.

Normal priority would be applied if two real time commands were simultaneously present in an IOC. The Real Time Command continues along with the related command to the receiving unit and any Response Command would be prefixed by the Real Time
Command to expedite the data return.

The Address Size 32, Address Size 40, Address Size 48 and
Address Size 64 Commands are used to set the size of the address that is to be used for data transfer commands. When a device is powered up, the interface is not required to be in any address mode. One of the Address Size commands is required before Read, Write, Exch or CmpSwp commands can be processed.

A device receiving an Address Size command will propagate that command to all of its downstream ports.

The Big Endian and Little Endian commands are used to establish the byte ordering of multiple byte data. For a device that only operates as a slave, these commands establish the byte ordering of multiple byte data read from or written to the device. For mastering devices, this command also lets the device know the order of data and addresses read from control blocks in memory. One of the Endian commands is required before Read, Write, Exch or CmpSwp commands can be processed.

A device receiving an Endian Polarity command will propagate that command to all of its downstream ports.

The Channel Reset command is used to restart a device or to clear a channel of any outstanding requests. A device receiving this command from an upstream port should clear its queue of ordered read requests and retransmit the command on all of its downstream ports.

The CLK signal provided by the upstream device will always be 50 MHz. Devices may have actual A-Net transfer clock rates of either25 MHz,50-MHz, or integer multiplesaf 50 MHz.

Devices with transfer rates of 25 MHz will internally divide the 50 MHz A-Net CLK to derive their 25 MHz transfer clock.

Devices with transfer rates of 50 MHz may use the A-Net CLK directly. Devices with transfer clock rates of integer multiples of 50 MHz will use clock multiplier circuitry (synchronized to CLK) to generate their internal transfer clocks. In these cases data values are actually transferred based on the internal clocks, the CLK signal just providing the synchronizing reference. A device will have the capability of transferring at a minimum of 50 MHz on any downstream port.

At system initialization, the 50 MHz channel clock (CLK) is generated for all A-Net devices. A delay from System Reset will elapse before channel initialization is initiated. This delay will provide adequate time for any phase-locked-loops in devices to stabilize. Each A-Net device will then enter an initialization phase. During initialization each A-Net device with a downstream connection will assert its RTS* line for one period of its maximum transfer rate. Within a predetermined number of cycles of the 50 MHz CLK, the downstream device will respond by asserting its RTS* for one period. The period of the response will be the longer of the period of the downstream device's maximum transfer rate or the period of the received pulse of the RTS* sent by the upstream device.

It is possible that the period of the RTS* sent by the upstream device for channel initialization may be of too short a duration to be "seen" by the downstream device. So, after the predetermined clock response period, the upstream device will step down its transfer clock rate by half (200 MHz to 100
MHz, 100 Mhz to 50 Mhz, 50 MHz to 25 MHz) and retry the initialization sequence until an initialization sequence at 25
MHz fails to produce a response. At this point, the upstream device will disable the CLK to the non-responding downstream device.

When an upstream device is communicating with a slower downstream device, the converse case is not possible, it is necessary to establish clock phase relationship for the divided clocks. For example, two devices that have "agreed" on a 25
MHz data transfer rate must insure that they are both making negative transitions on their divided clocks on the same negative edge of the 50 MHz CLK. The protocol defines that the upstream device will assert its RTS* on the rising edge of its transfer clock. Both CLK and RTS* are sent by the upstream device and arrive at the downstream device with minimal phase distortion. The downstream device may use the falling edge of
RTS* from the upstream device to establish the correct phase relationship.Once this phase relationship is established during channel initialization, there should be no need to continue to use the upstream's RTS*

to set the phase.

At transfer rates higher than 50 MHz, both the upstream and the downstream devices will use the 50 MHz CLK in their frequency multiplying circuitry. Since the duty cycle of the 50 MHz CLK is difficult to control accurately, the multiplying circuitry should only lock on the falling edge of CLK. If phase-locked loops are employed in the doubling circuitry, they should be low slew rate to minimize jitter. The CLK between devices should not be stopped when the transfer rates between devices exceed 50 MHz.

At high frequencies, the time of flight for the signals accounts for a significant amount of the total period of the transfer. At A-Net transfer rates of 100 and 200 MHz, the round trip delay time is too long to allow the channel to continue to operate synchronously without some compensation for the time of flight of the signals. This compensation is pr#ovided by requiring that upstream devices transferring at 100 MHz and 200 MHz will have CLKIN. CLKIN is CLK routed to the downstream device CLK pin and back to the upstream CLKIN pin.

The upstream device will use CLKIN to derive its receive clock.

If the upstream device is only capable of 50 MHz operation, CLKIN is not required and CLK may be terminated at the downstream device.

Systems using the protocol of the present invention can be designed to take advantage of stopping the A-Net CLK on one or more channels. Being able to stop the A-Net channel clock can reduce the power consumed by infrequently used I/O devices, or by any I/O device when the system is in hibernation-mode, and by channels with no devices to reduce RFI. Stopping clocks will also aid in "hot plugging" new I/O devices. When conditions described below are met to allow a channel clock to stop, the clock will freeze in the "pulled-up" state. If the IOC needs to access the I/O device it will merely restart the clock.If the I/O device needs to access the system (via anIOC), then it must activate its RTS* signal (asynchronous to theIOC) at least long enough for the IOC to sample the CTS signal through its metastable avoidance circuitry and get the channel clock started again.

There are several conditions that will allow a channel clock to stop during run-time. The devices must be fully static, or provide their own clock to keep internal registers alive. There must be an alternate means for an I/O device to activate its RTS* signal, without using the channel clock, when it needs to request access of the system while the clock is stopped. All devices downstream must be capable and willing to have their channel clock stopped. A timeout can be enabled for each channel that stops the clock after an arbitrarily chosen period of idleness. The default is preferably to have no timeout.

In hot-pluggable systems where I/O devices can be installed or replaced while the system is on, there must be a means for an I/O device to request that the channel become idle and undriven. This could be via an interrupt that tells the system to disconnect the I/O device. The I/O device cannot be disconnected and removed until all pending accesses to and from the device are satisfied or aborted.

When no device is installed, there will be no pullup on the channel clock line, and there will be nothing driving the CTS signal (other than a pull-up). Since the clock cannot transition high without a pull-up, the clock trace cannot radiate RFI even if the IOC output was still active. To conserve power, the IOC should sense that the clock changing and stop driving the clock and all other channel signals until the IOC CTS signal is sensed driven low by a device.

When a new device is plugged-in and ready to interface to the system, its RTS* signal is set active telling the IOC to interrupt the system and install the device.

Hibernation allows a system to go into a very low power mode and then wake up later in the same state that it was in before. There would be absolutely no I/O device activity allowed to the system and minimal activity to its external interfaces. The idea is that by stopping the clock, the system state will remain unchanged. The IOC will know whether the device can allow the clock to stop or not, and whether there are pending accesses that would need to be satisfied, so when hibernation mode finally starts, the clock would freeze the I/O devices in whatever state they happened to be in. There may be devices that are static and use only the channel clock for their internal state machines.This type of device can hibernate but cannot have its clock stopped during idle times, since stopping the clock would prevent remote activity

and there would be no way for the device to request that the clock be restarted.

Odd parity is used to detect errors in the transmission of packets across a channel. The use of parity is not intended to detect errors in other parts of communicating devices. The most common type of channel failures are expected to be stuckat failures and transient single bit failures. Parity, in combination with the A-Net protocol and signaling techniques, is designed to allow detection of either of these types of failures.

An odd number of stuck-at-zero or stuck-at-one conditions on the data/parity lines will be detected with a parity check by the receiver during normal transmission. All even (and odd) stuck-at-zero cases will be detected during a stuck-at-zero cycle. The stuck-at-zero cycle is defined to be the idle cycle immediately following a transmission. During this cycle, ports at both ends of a channel are required to turn off output drivers for the data/parity lines, allowing them to float high.

Sampling the data/parity lines on the clock following a stuckat-zero cycle by ports at both ends of the channel will reveal all stuck-at-zero conditions present on the data/parity lines at that time.

A stuck-at condition (zero or one) on a port's RTS* line will be detected by having each port sample its own RTS* line after every transition. If the sampled value is not equal to the value driven on the previous clock, an error condition exists.

Detection of a stuck-at-zero condition on a port's CTS line is accomplished by the port realizing that the CTS line has been held at a logical zero longer than the maximum packet length, which should not happen even during back-to-back transfers. The detection of a stuck-at-one condition on a port's CTS line, which keeps the port from receiving any messages, will be detected with a Read (or ReadS or Exch) timeout error.

A device detecting an error on a channel must signal the device on the other end that an error condition exists. This is accomplished by pulling all channel lines (except CLK) low for a predetermined number of 50 MHz clocks. After signaling an error condition, a device must wait until CTS and RTS* are high before proceeding. A device signaling an error on an upstream channel must broadcast a Channel Error message to its downstream channels. The device will then reset. All devices receiving a Channel Error message must broadcast that message to downstream ports and reset. Devices signaling an error on a particular downstream channel must send a Read Error message upstream for each outstanding Read Request (and ReadS and Exch) on that channel. These devices must then inform the system of the error.

A node receiving a Read Request is required to return some indication that the Read Request was received within a predetermined number of clocks. The purpose of this indication is to insure that the connection between devices is still operational and that some channel action by the receiver is still possible. This avoids the possibility of a long or infinite delay on a Read Response that might hang the system.

This indication can be provided in one of two ways. The first is by providing the requested data as a Read Response. If the accessed device cannot provide the Read Response within the predetermined number of clocks, it should perform some operation that involves assertion of it RTS*, such as issuing a message or providing a Request Retry, as described above.

A node sending a Read Request must maintain a Read Request
Timeout counter and a count of Read Requests (Read Request
Counter) issued on a channel. The Read Requests Counter is incremented when a ReadRequest, or. Compare and Swap, or
Exchange command is sent on a channel and decremented when a
Read Response or Read Error is received from that channel. Any channel activity by the port receiving the Read Response,
Request Retry, or Read Error will reset the Read Request
Timeout. The counter will continue to run after a reset as long as the Read Request Count is non-zero. In the event that the Read Request Timeout expires, the requesting node will issue Read Error messages (one for each pending read on the channel) back to the original requestor.

The functions of an A-Net Port (ANP) are to comprehend the channel protocol, normally by use of a state machine; to decode the received command byte for tracking purposes and to provide the command information to associated control logic; to encode packets in case of reset and errors; to check for parity